LAMP-TR-084 2002
CS-TR-4352
UMIACS-TR-2002-32

Creating Parsing Lexicons from Semantic Lexicons Automatically and Its Applications

Necip Fazil Ayan, Bonnie Dorr

Language and Media Processing Labratory
Instititue for Advanced Computer Studies
College Park, MD 20742

Abstract

In an earlier study, we described a method to create a parsing lexicon from semantic-based lexicon using manual rules. This paper describes an automated mapping methodology for the same purpose. Our approach maps lexical entries in a large LCS-based repository of semantically classified verbs to their corresponding syntactic patterns automatically. We evaluate the accuracy and coverage of this lexicon using LDOCE syntactic codes as a gold standard. We show that this lexicon is comparable to the hand-generated Minipar lexicon (i.e., similar recall and precision values). We also present the effects of using such a lexicon on the parser performance. The advantage of automating the process is that the same technique can be applied directly to lexicons we have for other languages, for example, Arabic, Chinese, and Spanish. The results indicate that our method will help us generate parsing lexicons which can be used by a broad-coverage parser that runs on different languages.

***The support of the LAMP Technical Report Series and the partial support of this research by the National Science Foundation under grant EIA0130422 and the Department of Defense under contract MDA9049-C6-1250 is gratefully acknowledged.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2002		**00-00-2002 to 00-00-2002**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Creating Parsing Lexicons from Semantic Lexicons Automatically and Its Applications	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Language and Media Processing Laboratory,Institute for Advanced Computer Studies,University of Maryland,College Park,MD,20742-3275**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**11**	

Creating Parsing Lexicons from Semantic Lexicons Automatically and Its Applications

Necip Fazıl Ayan and Bonnie J. Dorr
Department of Computer Science *and* UMIACS
University of Maryland
College Park, 20742, USA
{nfa, bonnie}@umiacs.umd.edu

Abstract

In an earlier study, we described a method to create a parsing lexicon from a semantic-based lexicon using manual rules. This paper describes an automated mapping methodology for the same purpose. Our approach maps lexical entries in a large LCS-based repository of semantically classified verbs to their corresponding syntactic patterns automatically. We evaluate the accuracy and coverage of this lexicon using LDOCE syntactic codes as a gold standard. We show that this lexicon is comparable to the hand-generated Minipar lexicon (i.e., similar recall and precision values). We also present the effects of using such a lexicon on the parser performance. The advantage of automating the process is that the same technique can be applied directly to lexicons we have for other languages, for example, Arabic, Chinese, and Spanish. The results indicate that our method will help us generate parsing lexicons which can be used by a broad-coverage parser that runs on different languages.

1 Introduction

This paper extends the technique described in an earlier work [1] for generating parsing lexicons for a principle-based parser (Minipar [7, 8]) using a lexicon that is semantically organized according to Lexical-Conceptual Structure (LCS) [2, 4]—an extended version of the verb classification system proposed by [6]. As opposed to the method in [1], this paper presents an automated technique to generate such a parsing lexicon.

Our goal is to see how much syntactic information we can extract from a semantic based lexicon. More specifically, our goal is to evaluate the coverage of such a parsing lexicon and then ultimately to measure the parser performance using the generated parsing lexicon. For the evaluations on coverage, we developed a mapping from the codes of Longman's Dictionary of Contemporary English (LDOCE [12])—the most comprehensive online dictionary for syntactic categorization—to a set of syntactic patterns. We use these patterns as our gold standard and show that our derived lexicon is comparable to the hand-generated Minipar lexicon (i.e., similar recall and precision values). For the evaluations on the parser performance, we run a broad-coverage parser (Minipar) based on two different lexicons and show the parser produces similar precision and recall values in terms of dependency links.

The main motivation behind the automation of such a mapping is the portability: We currently have LCS lexicons for English, Arabic, Spanish, and Chinese, so our automated approach allows us to produce parsing lexicons in each of these languages, using a small set of rules.

The remainder of the paper is organized as follows: Section 2 presents the background information: the description of the code sets used in this paper and the motivations of this work. Section 3 explains how the automated mapping technique works in detail. In Section 4, we give the experiment results on the coverage of the generated lexicons, and in Section 5 we

present the effects of the different lexicons on parser performance. Section 6 discusses whether the claim of generating syntactic lexicons from LCS-based lexicons in this paper is true. In the last section, we conclude the paper and give some future research directions.

2 Background Information

Our initial goal is evaluate the accuracy and coverage of a parsing lexicon where each verb is classified according to the arguments it takes. We use syntactic patterns as the basis of the comparison between our parsing lexicon and the original lexicon used in Minipar.

Syntactic patterns simply list the type of the arguments one by one, including the subject. Formally, a syntactic pattern is $a_1, a_2, \ldots$ where a_i is an element of NP, AP, PP, FIN, INF, BARE, ING, WH, PREP, corresponding to noun phrases, adjective phrases, prepositional phrases, clauses beginning with "that", infinitive clauses, bare infinitive clauses, "-ing" clauses, "-wh" clauses and prepositions, respectively. Prepositional phrases may be made more specific by including the heads, which is done by PP.*prep* where *prep* is the head of the prepositional phrase. The first item in the syntactic pattern gives the type of the subject.

Our initial attempts at comparing LCS-based lexicons involved the use of the OALD code set instead of syntactic patterns. We chose OALD code set because it is the code set employed by Minipar, the current parser we use in our applications and we would like to port our new lexicon into Minipar. This approach has two problems, which are closely related. First, using the class number and thematic grids as the basis of mapping from the LCS lexicon to OALD codes is a difficult task because of the high degree of ambiguity. For example, it is hard to choose among four OALD codes (Ln, La, Tn or Ia) for the thematic grid _th_pred, regardless of the Levin class. In general, the grid-to-OALD mapping is so ambiguous that maintaining consistency over the whole LCS lexicon is virtually impossible.

Secondly, even if we are able to find the correct OALD codes, it is not worth the effort because all that is needed for the parsing lexicon is the type and number of arguments that can follow the verb. For example, $Cn.n$ (as in *"appoint him king"*) and $Dn.n$ (as in *"give him a book"*) both correspond to two NPs following the verb, but the second NP is a direct object in the former case and an indirect object in the latter. Since the parser relies ultimately on syntactic patterns, not codes, we can eliminate this redundancy by mapping any verb in either of these two categories directly into the [NP.NP.NP] pattern. Thus, using syntactic patterns is sufficient for our purposes.

The LCS lexicon consists of verbs grouped into classes based on an adapted version of verb classes [6] along with the thematic grid representations (see [2, 4]). Our purpose is to find the set of syntactic patterns for each verb in LCS lexicon. In an earlier work, we proposed a hand-crafted mapping where the semantic class number and the thematic grid of the verbs determine the correct syntactic patterns [1]. In this paper, we propose a new mapping methodology, which is completely automatic and language independent.

To observe how effective the automatic mapping is, we will compare the resulting lexicon with a gold standard. In this paper, we use LDOCE dictionary [12] as our gold standard in the experiments. LDOCE has a more detailed code set than that of OALD (and hence Minipar). The codes include both arguments and modifiers. Moreover, prepositions are richly specified throughout the lexicon. The syntax of the codes is either *CN* or *CN-Prep*, where C is the initial letter of the verb sub-categorization (as in the generic OALD codes) and N is a number, which corresponds to different sets of arguments that can follow the verb. For example, T1-ON refers to verbs that are followed by a noun and a PP with the head *on*. The number of codes included in this set is 179.

We will compare our new lexicon with respect to current Minipar lexicon, using LDOCE as the goal standard. Minipar is based on the OALD code set, which is used in Oxford Advanced Learner's Dictionary, a.k.a OALD [9]. The verbs are categorized into 5 main groups: Intransitive verbs, transitive verbs, ditransitive verbs, complex transitive verbs, and linking verbs. Each code is of the form $Sa_1[.a_2]$ where S is the first letter of the verb categorization ($S \in \{I, T, D, C, L\}$

Thematic Role	Syntactic Patterns
particle	PREP
prop(...), mod-prop(...), info(...)	FIN or INF or ING or PP
all other role(...)	PP
th, exp, info	FIN or INF or ING or NP
prop	NP or ING or INF or FIN or BARE
pred	AP or NP
perc	[NP.ING] or [NP.BARE]
all other roles	NP

Table 1: Syntactic Patterns Corresponding to Thematic Roles

for the corresponding groups), and $a_1, a_2, \ldots$ are the argument types. If a code contains more than 1 argument, each arguments is listed serially. Possible argument types are n for nouns, f for finite clauses (*that* clauses), g for -ing clauses, t for infinitive clauses, w for finite clauses beginning with -wh questions, i for bare infinitive clauses, a for adjectives, p for prepositions and pr for prepositional phrases. The codes do not explicitly specify which prepositions can be used in the prepositional phrases. For example, Tn refers to the verbs followed by a noun ('She read the book'), $Tn.pr$ refers to the verbs followed by a noun and a prepositional phrase ('He opened the door with a latch'), and $Dn.n$ refers to the verbs followed by two nouns ('She taught the children French'). The number of codes in OALD code set is 32

3 Automatic Generation of Syntactic Patterns

The lexicon derived from the hand-crafted mapping between the LCS lexicon and the syntactic patterns is comparable to the current Minipar lexicon. However, the mapping required a great deal of human effort, since each semantic verb class must be examined by hand in order to identify appropriate syntactic patterns. The process is error-prone, laborious, and time-intensive (approximately 3-4 person-months).

In a recent experiment, we developed an automated mapping (in 2 person-weeks) that takes into account both semantic roles and some additional features stored in the LCS database, without reference to the class number. The additional features used in the mapping is stored in VARSPEC field of LCS lexicons.

3.1 Why Do We Need VARSPEC?

VARSPEC is used in LCS lexicons to specify some features of each thematic role for a verb. This information may specify that the role can be optional or obligatory, human or animal or anything, abstract or any concept, singular or plural, animated or all, and so on. In addition to those concepts, VARSPEC can specify the type of the argument. The roles are represented as unique numbers in VARSPEC, which can be found in [2]. For example, the verb "remind" in class 37.9.d.ii with the thematic grid _ag_goal_info(to) is given the following VARSPEC:

```
:VAR_SPEC ((1 (human +)) (6 (human +)) (2 (thing -) (cform inf)))
```

Table 1 summarizes the automated mapping rules. In the cases where the syntactic pattern is ambiguous and there is no specification for the verbs, default values are used for the mapping: BARE for "prop", AP for "pred" and NP for "perc".

This specification restricts the *agent* and the *goal* to be a human and *info* to be an infinitive clause.

During the generation of mapping from the thematic grids to the list of patterns, we are only interested in the type of the arguments specified in VARSPEC. The other features of VARSPEC do not provide useful information to determine the possible syntactic patterns.

If VARPSEC contains	Syntactic Pattern
(cform fin)	FIN
(cform inf)	INF
(cform fin/inf)	FIN and INF
(aspect prog)	ING
(cat a)	AP
(cat a/n)	NP and AP
nothing above	default values

Table 2: How To Solve Ambiguities Using VARPSEC

The motivation behind using VARSPEC as well as the thematic grids lies in the fact that the thematic grids are not sufficient to correctly identify the syntactic patterns for those verbs. Our algorithm to find the syntactic pattern corresponding to a thematic grid works iteratively, i.e. role by role in the thematic grid. For example, to find the patterns for _ag_th_prop, we first find the pattern for the *agent (ag)*, then for the *theme (th)* and finally for the *property (prop)*, and then combine all those patterns to have the pattern for the whole grid. Thus, we need the ability to find the pattern for each role in the thematic grid without looking at the other roles. The necessity of using VARPSEC comes into the picture at this point. We are not able to find the pattern for each role in the LCS database by knowing only that it is an *agent, theme, prop, pred* and so on. For example, the thematic role *prop* has five different syntactic patterns in different Levin classes: NP, FIN, INF, ING or BARE. Similarly, the thematic role *pred* can be either an AP or NP in different classes. For example, *pred* becomes an AP for the verb *behave* in class 29.6.a while it is mapped onto an NP for the verb *carry* in class 54.2.

One approach to solve these type of ambiguities is to look at the class number or specific verbs and determine the correct pattern for those thematic roles. However, this approach requires that a human should examine all Levin classes, which means lots of human effort for each language in consideration. We, on the other hand, would like to create a mapping scheme which is language independent with minimal human effort.

Another approach is to use the keywords encodes in the thematic roles to find the correct pattern. For example, we certainly know that *prop(that)* is a FIN, *mod-prop(to)* is an INF, and all patterns with a specific preposition *prep* between the parentheses in it is a PP.*prep* (examples include *info(about), goal(to)*, and so on.). Even though we implemented this approach in an early study, we realized that one of our objectives, language independence, is not satisfied. The results of the mapping are correct and satisfactory, however, the mapping algorithm is required to be modified for every new language. Certainly, we would like to avoid that repetition of work, by implementing a language independent mapping scheme.

VARPSEC field in the LCS database helps us to avoid much human effort and obtain a language-independent mapping. For the ambiguous thematic roles, VARPSEC specifies what kind of arguments can be used for those roles. For example, for the thematic role *prop*, the pattern is identified by the type of clause in VARPSEC for *prop*. In the case of the role *pred*, the category field in VARSPEC helps us to find the correct pattern for *pred*. Table 2 summarizes what kind of patterns must be used in case of ambiguity. The first column is the field in VARPSEC for the role in question and the second one gives the correct pattern.

If the VARPSEC for this role includes the specification in the first column, then the pattern in the second column is assigned to the role. (The only exception occurs with *perc* where NP is added to the beginning of the pattern in the second column of Table 2. In this case, *perc* corresponds to the list of two patterns: NP and one another.) If the VARPSEC for a specific role does not include any of the entries in Table 2, then the default patterns are assigned: PP for all the roles with parentheses (such as *mod-prop(), info()*, etc.), BARE for *prop*, AP for *pred*, and NP for all others.

Automated_Mapping(*LCS database*);
set syntactic patterns list of each verb to empty set
for each (*verb*, *grid*) pair in *LCS database* **do**
  for each *role* in the *grid* **do**
    find syntactic pattern for *role* according to Table 1 and Table 2
  find all possible realizations of *grid*
  generate all possible syntactic patterns for all realizations of *grid*
  add the resulting patterns to the list of patterns for *verb*

Figure 1: Automated Mapping Algorithm

3.2 Algorithm

Syntactic patterns for each thematic grid are computed by combining the results of the mapping from each thematic role in the grid to a syntactic pattern, one by one. The correct pattern for each thematic role is computed using Table 1 and Table 2.

An additional modification while dealing with PPs is the inclusion of specific prepositions to PP if the roles explicitly specifies it. For example, the thematic role *instr(with)* is mapped not to a PP but to a PP.with.

Most of the thematic grids in the LCS database contains optional roles: The verb either can take those optional roles or not in different contexts. If the grid includes optional roles, every possibility is explored and the syntactic patterns for each of them is included in the whole list of patterns for that grid. For example, the syntactic patterns for _ag_th,instr(with) include the patterns for both _ag_th and _ag_th_instr(with), which are [NP.NP] and [NP.NP.PP.with]. With this technique, if there are n optional roles in the thematic grid, at most 2^n different syntactic patterns may be produced for this thematic grid. The number of different patterns may be less than 2^n because of repetitions of patterns. For example, _ag_th,src(),goal() will produce only two syntactic patterns: [NP.NP] and [NP.NP.PP].

A sketch of the algorithm is given in Figure 1. For each entry in the LCS database (i.e. verb+grid), we find the syntactic pattern for that pair of verb and thematic grid, as described above. Then, the algorithm combines all the patterns for each verb to produce a list of verbs and possible syntactic patterns for that verb. The final list will be referred to LCS-based lexicon in the experiments.

Note that our mapping algorithm eliminates the need for using the same syntactic patterns for all the verbs in a specific class: Verbs in the same class can be assigned different syntactic patterns with the help of additional features in the database. Thus, we need not rely on the semantic class number at all during this mapping. Moreover, the mapping can be extended to other languages without incorporating any information about those languages. Simply, the mapping scheme described here covers all possible languages we have an LCS database.

4 Experiments on Coverage

Similar to the mapping applied to LCS database, each LDOCE code was mapped manually to one or more patterns. LDOCE codes are more refined than the generic OALD codes, but mapping each to syntactic patterns provides an equivalent mediating representation for comparison. For example, LDOCE codes D1-AT and T1-AT are mapped onto [NP.NP.PP.at] by our mapping technique. Again, this is a many-to-many mapping but only a small set of LDOCE codes map to more than one syntactic pattern. As a result of this mapping, we produced a new lexicon from LDOCE entries, similar to LCS-based lexicon . We will refer to this lexicon as the LDOCE-based lexicon below.

To measure the effectiveness of our mapping from LCS entries to syntactic patterns, we compared the precision and recall our derived LCS-based syntactic patterns with the precision

	Minipar Lexicon (All verbs in Minipar Lexicon)	**Minipar Lexicon (Common verbs with LCS Lexicon)**	**LCS Lexicon**	**Intersection of Minipar and LCS Lexicons**
Precision	80%	81%	61%	91%
Enhanced Precision	81%	82%	80%	91%
Recall	68%	67%	61%	50%

Table 3: Precision and Recall Summary: Minipar- and LCS-based Lexicons

and recall of Minipar-based syntactic patterns, using LDOCE-based syntactic patterns as our "gold standard".

Each of the three lexicons contains verbs along with their associated syntactic patterns. For experimental purposes, we convert these into pairs. Formally, if a verb v is listed with the patterns $p_1, p_2, \ldots$, we create pairs (v, p_1), (v, p_2) and so on. In addition, we have made the following adjustments to the lexicons, where L is the lexicon under consideration (Minipar or LCS):

1. Given that the number of verbs in each of the two lexicons is different and that neither one completely covers the other, we take only those verbs that occur in both L and LDOCE, for each L, while measuring precision and recall.
2. In the LDOCE-based lexicon, the number of arguments is never greater than 2. Thus, for a fair comparison, we converted the LCS-based lexicon into the same format. For this purpose, we simply omit the arguments after the second one if the pattern contains more than two arguments/modifiers.
3. The prepositions are usually not specified in LDOCE-based lexicon. Thus, we ignore the heads of the prepositions in LCS-based lexicon, i.e., if the pattern includes [PP.*prep*] we take it as a [PP].

Precision and recall are based on the following inputs:

A = Number of pairs in L occurring in LDOCE
B = Number of pairs in L NOT occurring in LDOCE
C = Number of pairs in LDOCE NOT occurring in L

That is, given a syntactic pattern encoded lexicon L, we compute:

(1) The *precision* of $L = \frac{A}{A+B}$;
(2) The *recall* of $L = \frac{A}{A+C}$.

In our experiments, L stands for either Minipar-based lexicon or LCS-based lexicon. In the early study [1], we compared the lexicon created by hand-crafted mapping with Minipar-based lexicon. The results are summarized in Table 3. The last column in Table 3 give the precision and recall values for the intersection of the syntactic patterns in Minipar-based and LCS-based lexicons for each verb. For Minipar-based lexicon, we achieve 80% precision as opposed to a 61% precision for LCS-based lexicon. Similarly, the recall is better for Minipar-based lexicon, 68% as opposed to a 61% recall for LCS-based lexicon. As expected, the precision of the intersection lexicon is high and the recall is low.

In this paper, we concentrate on the precision/recall values achieved by the automated mapping scheme. This experiment resulted in a lexicon that is comparable to the manually generated LCS-based lexicon above—with the precision/recall results shown in Table 4.

We found that the precision was higher for Minipar than for the LCS lexicon, but when we examined this in more detail, we found that this was almost entirely due to "double counting" of entries with optional modifiers in the LCS-based lexicon. For example, the single LCS-based grid _ag_th,instr(with) corresponds to two OALD+ codes, [NP.NP] and [NP.NP.PP] while

Verbs in LDOCE Lexicon	5648
Verbs in LCS Lexicon	4267
Common verbs in LCS and LDOCE	3757
Pairs in LCS Lexicon	9253
Pairs in LDOCE Lexicon	9200
Pairs in LCS and LDOCE	5634
Verbs fetched completely	1781
Precision	61%
Enhanced Precision	80%
Recall	61%

Table 4: Precision and Recall of Automatic Generation of Syntactic Patterns

Trial Type (C/D)	Number of	OALD-based			LCS-based		
C/D	sentences	Pr	Rc	Rt	Pr	Rc	Rt
Y/+	1715	62	59	58	64	61	60
N/+	1544	63	59	58	64	61	60
Y/-	1053	63	60	58	64	61	60
N/-	948	63	60	59	64	61	60

Table 5: Parser Performance Using Generated Lexicon

LDOCE takes this only as [NP.NP]. Specifically, 53% of the non-matching LCS-based patterns are [NP.NP.PP]—and 93% of these co-occur with [NP.NP]. Similarly, 13% of the non-matching LCS-based patterns are pattern [NP.PP]—and 80% of these co-occur with [NP].

This is a significant finding, as it reveals that our precision is spuriously low in our comparison with the "gold standard." In effect, we should be counting the LCS-based pattern [NP.NP.PP]/[NP.NP] to be a match against the LDOCE-based pattern [NP.NP]—which is a fairer comparison since neither LDOCE nor Minipar takes modifiers into account. (We henceforth refer to LCS-based the co-occurring patterns [NP.NP.PP]/[NP.NP] and [NP.PP]/[NP] as overlapping pairs.) To observe the degree of the impact of optional modifiers, we computed another precision value for the LCS-based lexicon by counting overlapping patterns once instead of twice. With this methodology, we achieved 80% (enhanced) precision. This is equivalent to the enhanced precision we got for LCS-based lexicon with hand-crafted mapping.

5 Experiments on Parser Performance

The experiments showed that the coverage of the parser is as good as the results obtained using the current Minipar lexicon. However, the ultimate goal of this study is to create lexicons that can be embedded into a parser. Thus, we have to measure the effects of these generated lexicons on the parser performance. We will use Minipar as the parser in our evaluations because we can easily embed a different lexicon to Minipar and it is one of the best parsers in English in the literature. In the following, we will compare the performance of Minipar based on two different lexicons in English: one currently used by Minipar, which is hand-crafted for parsing purposes, and one we generated from LCS-based lexicon using the automated mapping.

Table 5 shows the results when we embed the generated lexicon into Minipar as well as the results of using the current Minipar lexicon. The parser is run on Section 09 of the Wall Street Journal. In the first column of Table 5, "C" refers to whether the sentences contain verbs that sub-categorize for clausal arguments, and "D" refers to whether the sentences contain non-Levin (LDOCE) verbs. In the third and fourth column, "Pr" corresponds to the average precision (in

dependency links) per sentence, "Rc" corresponds to the average recall (in dependency links) per sentence, and "Rt" corresponds to the recall on the whole "bag" of sentences.

As Table 5 shows clearly, using the generated lexicons instead of the original one nearly gives the same performance on the parser in terms of average precision and recall in dependency links. Both lexicons will result in 62-64% precision value and 58-60% recall value. The original Minipar lexicon still produces 1-2% better performance with respect to the lexicon generated from LCS databases but the results are promising. The table clearly shows that the syntactic information extracted from a semantic based lexicon is nearly as good as the information in the lexicon which is specifically tailored for parsing purposes.

6 Discussion

The knowledgeable reader may question the mapping of a Levin-style lexicon into syntactic codes, given that Levin's original proposal is to investigate verb meaning through examination of syntactic patterns, or *alternations*, in the first place. There are several ways in which this database has become more than just a "semantified" version of a syntactic framework; we elaborate on this further here.

Levin's original framework omitted a large number of verbs—and verb senses for existing Levin verbs—which we added to the database by semi-automatic techniques. Her original framework contained 3024 verbs in 192 classes numbering between 9.1 and 57—a total of 4186 verb entries. These were grouped together primarily by means of syntactic alternations. Our augmented database contains 4432 verbs in 492 classes with more specific numbering (e.g., "51.3.2.a.ii") including additional class numbers for new classes that Levin did not include in her work (between 000 and 026)—a total of 9844 verb entries. These were categorized according to semantic information (using WordNet synsets coupled with syntactic filtering) [3]—not syntactic alternations.

An example of an entry that we added to the database is the verb *oblige*. We have assigned a semantic representation and thematic grid to this verb, creating a new class 002—which we call *Coerce Verbs*—corresponding to verbs whose underlying meaning corresponds to "force to act". Because Levin's repository omits verbs taking clausal complements, several other verbs with a similar meaning fell into this class (e.g., *coerce*, *compel*, *persuade*) including some that were already included in the original system, but not in this class (e.g., *ask*). Thus, the LCS Database contains 50% more verbs and twice as many verb entries since the original framework of Levin. The result is that we can now parse constructions such as *She compelled him to eat* and *She asked him to eat*, which would not have been analyzable had we compiled our parsing lexicon on the basis of Levin's classes alone.

Levin's original proposal also does not contain semantic representations or thematic grids. When we built the LCS database, we examined each verb class carefully by hand to determine the underlying components of meaning unifying the members of that class. For example, the LCS representation that we generated for verbs in the *put* class includes components of meaning corresponding to "spatial placement in some manner," thus covering *dangle*, *hang*, *suspend*, etc.

From these hand-generated LCS representations, we derived our thematic grids—the same ones that are mapped onto our syntactic patterns. For example, position 1 (the highest leftmost argument in the LCS) is always mapped into the agent role of the thematic grid. The grids are organized into a thematic hierarchy that provides the basis for determining argument assignments, thus enhancing the generation process in ways that could not have been done previously with Levin's classes alone—e.g., producing constructions like *John sent a book to Paul* instead of constructions like *The book sent John to Paul*. Although the value of the thematic hierarchy seems most relevant to generation, the overall semantic/thematic hierarchical organization enables the automatic construction of lexicons that are equally suitable for both parsing and generation, thus reducing our overall lexical acquisition effort for both processes.

Beyond the above considerations, the granularity of the original Levin framework also was not adequate for our interlingual MT and lexical acquisition efforts. Our augmented form of this

repository has brought about a more refined classification in which we are able to accommodate aspectual distinctions. We encode knowledge about aspectual features (e.g., *telicity*) in our LCS representations, thus sub-dividing the classes into more specific sub-classes. The tests used for this sub-division are purely *semantic* in nature, not syntactic. An example is the Dowty-style test "*He was X-ing* entails *He has X-ed*" [5], where X is atelic (as in *run*) only if this entailment is considered valid by a human—and telic otherwise (as in *win*).

The inclusion of this type of knowledge allows us to refine Levin's classification significantly. An example is Class 35.6—*Ferret Verbs*: In Levin's original framework, this class conflated verbs occurring in different aspectual categories. Using the semantic tests above, we found that, in fact, these verbs should be divided as follows [10]:

Ferret Verbs: nose ferret tease (telic); seek (atelic)

The implication of this division for parsing is that the verbal arguments are constrained in a way that was not available to us in the original Levin-style classification—thus easing the job of the parser in choosing attachment points:

Telic:

*He ferreted the truth from him.

He ferreted the truth out of him

Atelic:

He sought the truth from him.

*He sought the truth out of him

Finally, Levin makes no claims as to the applicability of the English classes to other languages. Orienting our LCS database more toward semantic (aspectual) features rather than syntactic alternations has brought us closer to an interlingual representation that has now been demonstrably ported (quickly) to multiple languages including Arabic, Chinese, and Spanish. For example, telicity has been shown to be a crucial deciding feature in translating between divergence languages [11], as in the translation of English *run across* as Spanish *cruzar corriendo*.

7 Conclusion and Future Work

In this paper, we created an automatic mapping mechanism which achieves a nearly equivalent precision value to the value for current Minipar lexicon and to the value for LCS-based lexicon with hand-crafted mapping. Thus, our approach demonstrates that examination of thematic-role and featural information in the LCS-based lexicon is sufficient for executing this mapping automatically. Automating our approach gives us the flexibility of re-running the program if the structure of the database changes (e.g., an LCS representation is modified or class membership changes) and of porting to a new language with minimal effort.

We are currently working on the evaluation of parsing lexicons in languages other than English. The languages we are working on are Chinese and Spanish. We generated the parsing lexicons in those two languages but we do not have sufficient tools to evaluate the coverage or parser performance of those lexicons. The current research direction is to extend Minipar to run in other languages and to convert the treebanks in those languages to dependency trees in order to measure the parser performance. We believe that the generated lexicons in other languages will also present good coverage and good parser performance.

References

[1] Necip Fazil Ayan and Bonnie J. Dorr. Generating A Parsing Lexicon from an LCS-Based Lexicon. In *Proceedings of the Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language DataComputational Lexicography for Natural Language Processing, LREC'2002.* Las Palmas, Canary Islands, Spain, 2002.

[2] Bonnie J. Dorr. *Machine Translation: A View from the Lexicon.* The MIT Press, Cambridge, MA, 1993.

[3] Bonnie J. Dorr. Large-Scale Dictionary Construction for Foreign Language Tutoring and Interlingual Machine Translation. *Machine Translation*, 12(4):271–322, 1997.

[4] Bonnie J. Dorr. LCS Verb Database. Technical Report Online Software Database, University of Maryland, College Park, MD, 2001. http://www.umiacs.umd.edu/~bonnie/LCS_Database_Documentation.html.

[5] David Dowty. *Word Meaning in Montague Grammar.* Reidel, Dordrecht, 1979.

[6] Beth Levin. *English Verb Classes and Alternations: A Preliminary Investigation.* University of Chicago Press, Chicago, IL, 1993.

[7] Dekang Lin. Principle-Based Parsing without Overgeneration. In *Proceedings of ACL-93*, pages 112–120, Columbus, Ohio, 1993.

[8] Dekang Lin. Dependency-Based Evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems, First International Conference on Language Resources and Evaluation*, Granada, Spain, May 1998.

[9] R. Mitten. *Computer-Usable Version of Oxford Advanced Learner's Dictionary of Current English.* Oxford Text Archive, 1992.

[10] Mari Broman Olsen, Bonnie J. Dorr, and Scott C. Thomas. Toward Compact Monotonically Compositional Interlingua Using Lexical Aspect. In *Proceedings of the Workshop on Interlinguas in MT, MT Summit, New Mexico State University Technical Report MCCS-97-314*, pages 33–44, San Diego, CA, October 1997. Also available as UMIACS-TR-97-86, LAMP-TR-012, CS-TR-3858, University of Maryland.

[11] Mari Broman Olsen, Bonnie J. Dorr, and Scott C. Thomas. Enhancing Automatic Acquisition of Thematic Structure in a Large-Scale Lexicon for Mandarin Chinese. In *Proceedings of the Third Conference of the Association for Machine Translation in the Americas, AMTA-98, in Lecture Notes in Artificial Intelligence, 1529*, pages 41–50, Langhorne, PA, October 28–31 1998.

[12] P. Procter. *Longman Dictionary of Contemporary English.* Longman, London, 1978.